13 | 性能测试的工程集成：如何与产品开发和运维业务有机集成？

2019-12-25 庄振运

性能工程高手课 进入课程 >

讲述：庄振运
时长 11:25　大小 10.46M

你好，我是庄振运。

前面几讲，我们讨论了性能测试的几个方面，包括测试的种类、测试的规划、工具以及经验和教训。

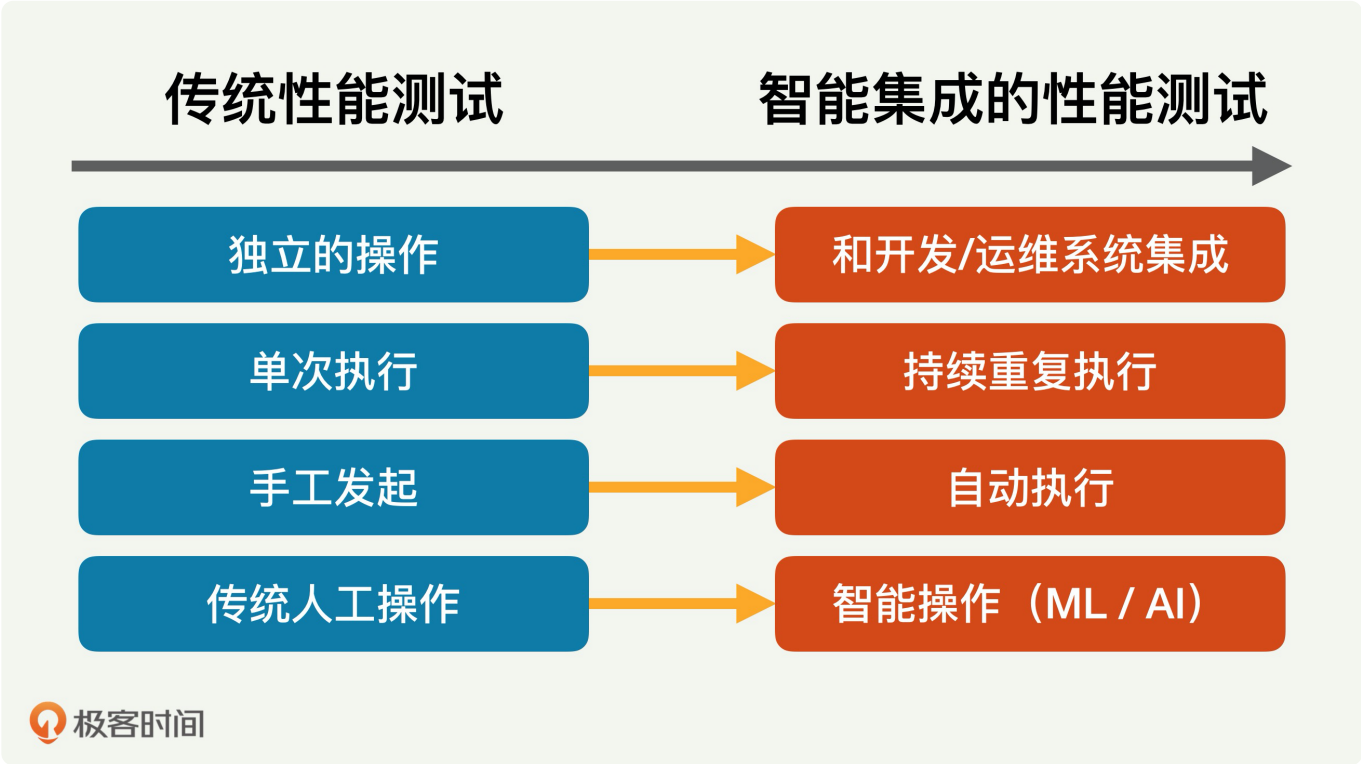
今天我们讨论性能测试如何和其他系统进行智能集成，也就是如何让**性能测试**这一工作从单独的、一次性的、手工发起的、传统的人工操作，进化成一个和开发及运维过程相结合的、持续的、自动重复执行的智能操作。

性能测试模式的演化

性能测试作为 IT 公司的一种重要工作，它的工作模式正在从传统的手工模式，不断进化成智能集成的自动模式。

这样的演化主要归功于这些年互联网技术的进步和业务需求的提高，包括数据量的加大和业务的日益复杂化、客户需求的多元化、公司业务规模的扩大，以及人工智能和机器学习的不断成熟。

那么，性能测试的模式进化表现在哪些方面呢？主要有四个方面，如下图所示：



1. 从**独立的操作**演化成和其他系统（比如开发和运维）的**有机集成**。

公司中很多业务都和性能测试有关，尤其是产品开发和系统运维业务。性能测试需要和这些业务紧密结合，从而使相关工作的效率极大提高。

2. 从一次次的**单独执行**演化成**持续而重复的执行**。

从开发角度来看，一个产品的程序代码会不断被开发和增强，包括加入新的功能和修补发现的错误。每次代码改变都需要进行性能测试，以确保程序面对客户的端到端性能和资源利用效率没有变低。从运维角度来看，公司的产品系统也需要持续地进行性能测试，以尽早发现可能的问题。

3. 从**手工进行**测试演化成**自动化**的测试。

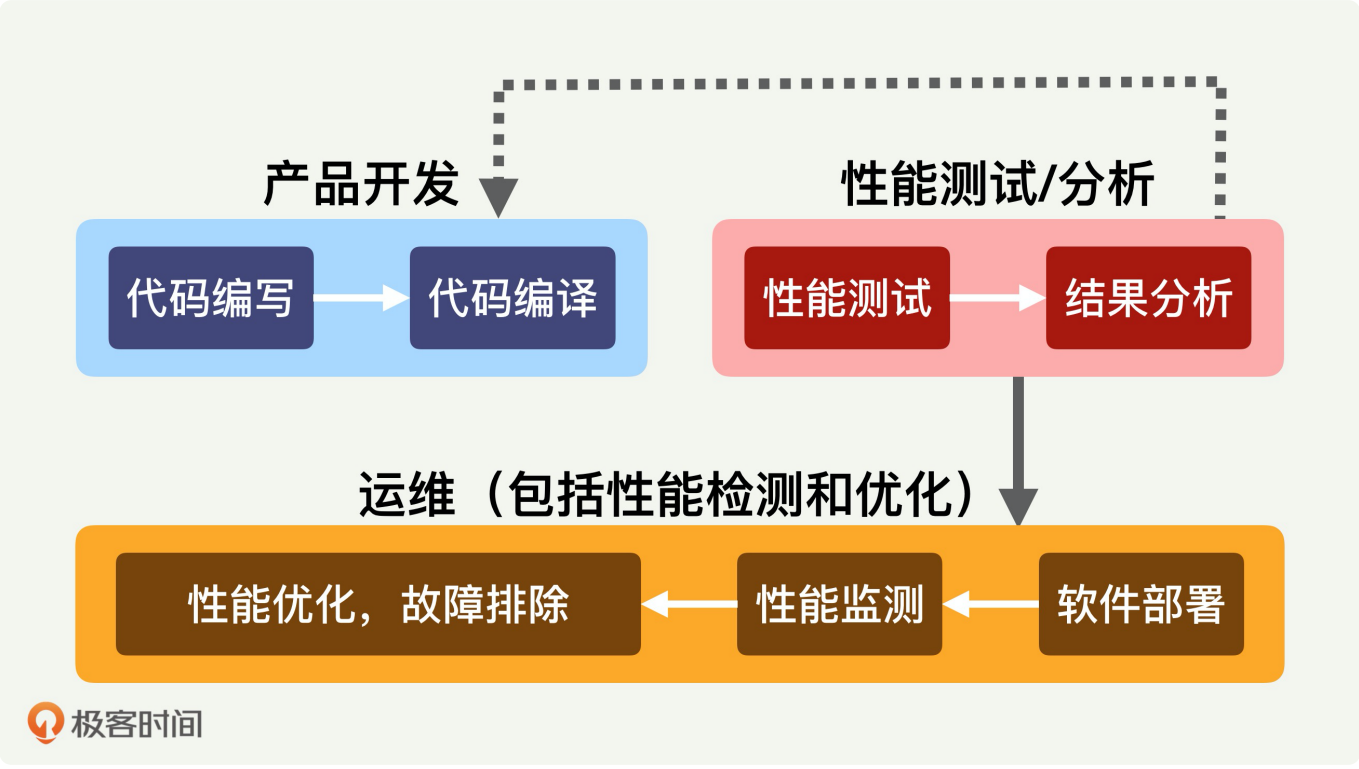
每当需要进行性能测试的时候，系统越来越需要自动触发，并按照既定规则来开始测试。测试完成后也需要自动进行分析，并根据分析的结果继续进行后面的定义好的步骤。整个过程最好不需要测试人员的人工参与，以降低运营成本并减少人为错误。

4. 从常规的**人工操作**演化成基于人工智能（AI）和机器学习（ML）的**智能操作**。

数据量的变大让有效的机器学习成为可能。同时伴随着人工智能技术的不断成熟，传统常规的性能测试一步步地走向智能的自动性能优化。

和产品开发的系统集成

性能测试和产品开发密切相关，我用下面这张图片表示它们之间的关系。我分成两部分讲，先讲和产品开发的系统集成。



性能测试和产品开发的集成也是所谓“持续集成”（Continuous Integration）的一部分。

什么是“持续集成”呢？

这个概念已经在软件开发领域存在很多年了，本来是为了配合敏捷开发（Agile Developement）的速度和效率而产生的，是把源代码管理、代码检查、程序编译、性能和功能测试、产品部署等一系列操作整合在一起的概念和工具。

怎么才叫“持续”呢?

就是从程序员提交了源码开始，相关工具就会自动进行编译、测试等一系列运作，并将结果反馈给程序员。这样，提交代码的程序员很快就会知道刚刚提交的代码有没有问题。

可能发生的问题有很多种，包括编译错误、整个程序不能运作、能编译运行但是整体性能变差等等。如果发现这样的问题，程序员和团队就可以迅速进行改正，不必等到开发周期后期才寻找和修复缺陷。

什么叫“集成”？

就是上述一套操作都是有相应工具支持的，几个工具又集成在一起，构成一个完整的系统。

持续集成包括两个重要阶段：持续交付和持续部署。我们先说一下持续交付。

持续交付（Continuous delivery）指的是开发人员的每一次代码提交，都会自动地编译，并且运行已经定义好的测试程序。

测试程序里面就包括性能测试和结果分析。如果分析的结果确认这次代码提交没有性能问题，就成功接受这次提交。否则，就会采取措施通知代码提交者去修正代码；并重复这个过程，直到通过。

持续部署（Continuous deployment）是持续交付的下一步，指的是代码通过评审以后，自动部署到真正的生产环境。

持续部署的目标是，代码在任何时刻都是可部署的，并且是可以进入生产阶段的。持续部署的前提是能自动化完成测试、构建、部署等步骤。同时，如果一旦部署了的版本发生不能接受的问题，就需要回滚到上一个版本。这个回滚的过程也需要简单方便，否则会造成非常大的混乱。

持续集成的价值何在?

持续集成的价值主要有几点：降低代码开发风险，及早发现集成错误，减少手动测试过程，快速生成测试结果，提高程序员和开发团队的安全感。同时，频繁的提交代码，也会鼓励和促使开发人员创建模块化和低复杂性的代码。

工具集

支持持续集成的工具有很多，按照领域分可以有下面几大类：代码版本管理、代码检查、编译连接、功能测试、性能测试、性能分析、代码覆盖率、结果展示等。有些流行的持续集成服务器可以整合这些工具或功能，比如 Jenkins、CruiseControl、Travis 等。

我举一个完整持续集成的工具例子。比如一个 Java 开发团队，在开发某项目时用了如下的一系列工具：

- 用 JStyle 来分析 Java 源代码
- 用 Ant 构建运行 JUnit 来进行简单的功能测试
- 用 DbUnit 执行长时间的数据库组件测试
- 用 JUnitPerf 来进行负载和压力测试
- 用 JProfiler 来进行全功能的代码性能剖析
- 用 Selenium 运行基于 Web 的功能测试
- 用 Cobertura 测量代码覆盖率
- 用 CruiseControl 作为服务器来管理持续集成

和运维业务的系统集成

讲完了性能测试和开发过程的集成，我们接着谈谈和运维业务的系统集成。

广义上来讲，性能工作也是运维的一部分，包括性能测试、性能分析和性能优化。但是，如果把性能工作和其他运维业务分开来看，它就需要和其他运维业务有机而智能地集成。

运维业务的一个趋势是智能化。

智能化的原因，是互联网数据量的变大，运维业务的多样化和复杂化，以及对运维服务质量要求的提高（比如低成本、低延迟、高防范）。这样一来，很多传统的运维技术和解决方案已经不能满足当前运维所需。另一方面，机器学习（ML）和 人工智能（AI）技术在飞速发展，这就推生了智能运维 AIOps（Artificial Intelligence for IT Operations），这是运维未来发展的必然趋势。

在这样一个趋势里，运维就需要和性能测试的过程紧密集成。

一是通过持续而智能的性能测试，能及时发现已有的和将来可能会发生的性能问题，从而快速修复和及时预防。比如，根据性能测试的结果，可能会发现在不久的将来，整个系统的某项资源会用光，有可能导致系统挂掉。这种情况，我们就可以提前采取相应的措施，来避免这一问题的发生。

二是通过不同种类的性能测试，来找出最佳的解决方案。或许，对有些简单的性能问题，我们能很容易发现和解决，但对很多复杂的性能问题，就比较难找出原因和确定最优方案。要找出最主要的性能问题根因，经常需要进一步的性能测试。即使找到根因，现代互联网服务产品的子模块之间依存度很高，互相的交互多而复杂。那么什么样的解决方案才是效果最好，最容易实现的的，往往需要进行进一步测试来验证。

性能测试和运维的集成必须有两个特点：**有机**和**自动**。

不管是性能衡量、问题预测、根因分析还是性能优化，人工去执行都非常费时费力，从而不可取。唯一可行的就是借助于人工智能来尽量自动化。除了持续自动地进行性能测试，发现性能问题还需要进行自动分析，找出问题后也要执行自动调整优化。

我们的目标是**让性能测试和运维业务进行智能的有机集成**，其中的智能来自于对大数据的分析和机器学习。

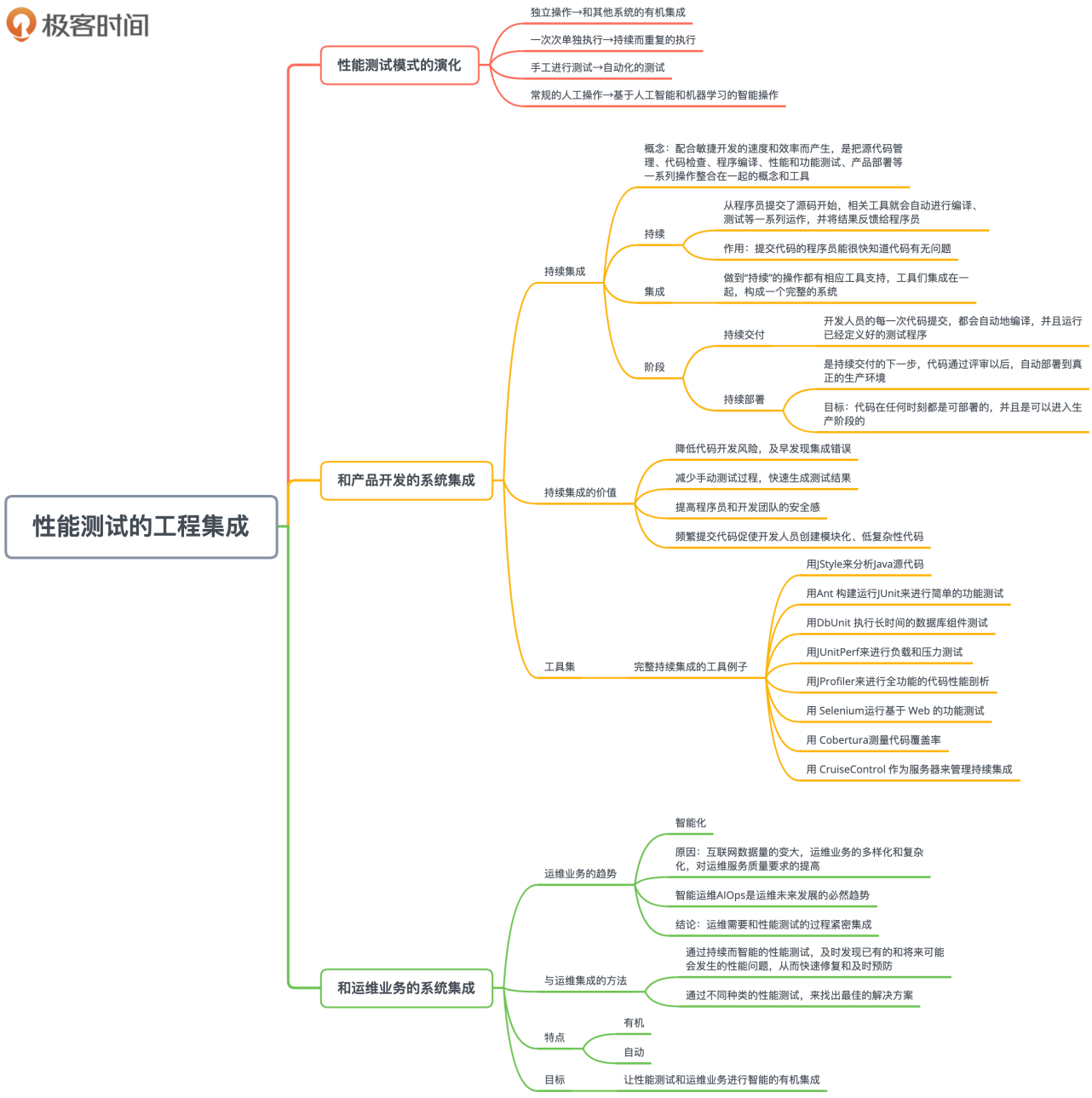
无论是本业务系统的历史数据，还是其他业务系统的数据，甚至是业界其他公司的数据和经验，都是机器学习的对象和分析的基础。同时，我们还需要注入适当的知识和规则，来帮助这一套集成的持续优化。

在这一过程中，数据的采集和整理是一切的基础。公司层面需要全方位，实时、多维度、全量地对各种运维数据采集、整理和存储。这里的运维数据包括基础架构的机器监控数据，内

网和外网的网络数据，公司业务流量数据，工单系统数据，日志监控数据等。这些数据需要有统一而合理的接口，以方便访问。

总结

性能测试虽然有很多讲究和注意的地方，但它本身也是作为整个公司业务的一个子系统而存在的。我们需要把它和其他几个子系统，尤其是产品开发子系统和运维子系统有机地整合集成起来，让这几个子系统之间的操作“浑然天成”，才能获取整个公司业务的最大收益。



白居易的《长恨歌》有两句：“在天愿作比翼鸟，在地愿为连理枝”，期盼的是一种比翼齐飞，一种同心同德。

这种期盼用在这一讲的子系统集成也挺合适。性能测试和产品开发子系统的集成，可以使得开发过程的迭代更快、更高效，并保证代码质量。性能测试和运维子系统的集成，可以让整个程序和业务保持高性能运转，提高公司业务质量和公司营收。

思考题

你现在的开发环境中有没有整合这样的自动测试系统？如果没有，你觉得值得考虑一下吗？如果你能从无到有的搭建一个这样的系统，你老板会不会对你刮目相看？

欢迎你在留言区分享自己的思考，与我和其他同学一起讨论，也欢迎你把文章分享给自己的朋友。

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇　12 | 九条性能测试的经验和教训：如何保证测试结果可靠且可重复？

下一篇　14 | 性能分析概述：性能问题归根结底是什么原因？

精选留言

写留言

> 由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。